

J’ai hacké ma clarinette !

GNU/Linux Magazine
Mois de parution
septembre 2018

Jarrige Grégory



Résumé
Vous avez dans un placard une vieille clarinette qui se morfond dans sa boîte. Un beau jour, l’envie vous prend de la ressortir, et de la connecter à tout ce qui peut produire du son électronique, et en premier lieu à votre ordi. Problème… comment connecter cet instrument à vent acoustique à la lutherie électronique du XXIème siècle ? La petite histoire que je vais vous compter peut s’appliquer à tous les instruments à vent.

Body

Je crois utile de préciser que je ne travaille pour aucun des fabricants que je vais citer dans cet article. Je jure que je n’ai touché aucune somme d’argent, ni aucun avantage en nature. Donc pas de conflit d’intérêts, juste de la curiosité et du fun.

La clarinette que j’utilise pour mes expériences est une bonne vieille clarinette soprano si bémol. Un modèle tout à fait classique, fabriqué il y a une trentaine d’années par une marque française tout à fait respectable. Si vous ne connaissez pas bien cet instrument, je vous invite à jeter un coup d’œil à la page **Wikipédia [1]** qui lui est consacrée et qui est très complète. Je vous avoue que j’ai délaissé cet instrument pendant près de 20 ans, alors les retrouvailles ont été un peu douloureuses, car il faut pas mal d’entraînement pour retrouver ses marques, ce n’est pas aussi simple que de remonter sur une bicyclette.

Mon objectif prioritaire – et l’objet de cet article - c’est de hacker le son de la clarinette avec tous les bidules électriques et électroniques qui peuvent me tomber sous la main.

Et donc se pose nécessairement le problème de la connexion électronique. Mon instrument n’est doté d’aucun orifice apparenté à du MIDI **[2]** ou de l’USB… *so*

what ! La solution la plus logique semble être le micro. Ceux qui s'y sont essayés le savent bien, la captation du son d'un instrument à vent n'est pas une sinécure. Où placer le micro, à quelle distance de l'instrument ? Pour obtenir un son correct, il faut du matériel de prise de son de bonne qualité, donc assez coûteux. Très important, il faut aussi avoir une pièce avec une bonne acoustique. Certains bidouilleurs ont tenté de fixer un micro piézo-électrique sur la paroi externe d'une clarinette, mais le son obtenu est assez éloigné du son réel de l'instrument.

Bref, j'en étais là de mes réflexions, quand un beau jour le Graal m'est apparu.

1. Le micro qui fait la différence

La solution à mon problème existentiel m’est apparue dans une vidéo du youtubeur Brian Hayes **[3]**. Dans cette vidéo, Brian présente un drôle de petit boîtier cylindrique, fixé sur le barillet d’une clarinette.

Le barillet, pour info, c’est une partie démontable de forme tubulaire, d’environ 6,5 cm, qui se trouve juste sous le bec. Sur la vidéo de Brian, on voit que le barillet est percé à mi-hauteur, une vis est fixée dans l’orifice, et le drôle de boîtier cylindrique est carrément vissé sur le barillet. Ce drôle de boîtier, c’est un micro piézo-électrique, le **Piezobarrel** (pour info, « barillet » se dit en anglais « barrel »). Grâce à ce système astucieux, le micro se trouve à l’intérieur de la clarinette, en contact direct avec la colonne d’air de l’instrument. Le micro capte donc le son au plus près, sans interférences extérieures.

Dans une seconde vidéo **[4]**, Brian Hayes nous fait écouter le son de la clarinette, capté par le Piezobarrel et transformé par une pédale d’effet pour guitare électrique. Le résultat est saisissant ! Dans une autre vidéo encore **[5]**, Brian montre le Piezobarrel fixé cette fois sur le bec d’un saxophone, et il nous fait écouter le son produit par le Piezobarrel branché sur différentes pédales d’effet (octaver, harmonist, delay).

À ce stade, autant vous dire que je suis mûr. Il me faut absolument ce Piezobarrel ! Recherche fébrile sur le web, ça coûte combien ce truc ? Je trouve le site officiel du fabricant **[6]**, qui est basé en Australie, découvre qu’il vend sa production sur un site d’enchères bien connu, ok… clic : bon, le fabricant propose plusieurs kits ? Je sélectionne celui qui est fourni avec un barillet pré-équipé d’une vis, sur laquelle il suffit de visser le Piezobarrel. Il coûte 134 $ australiens, soit environ 84 euros (à l’heure où je rédige cet article). Je passe commande, et j’attends une dizaine de jours avant de recevoir cette petite merveille (c’était juste avant Noël).

Sur la figure 1 vous pouvez voir la photo du kit que j’ai reçu.



piezobarrel

Fig. 1 : Le kit Piezobarrel.

J’aurais pu éventuellement acheter un kit sans barillet, mais le kit avec barillet ne coûtait pas beaucoup plus cher… Et puis zut, j’ai envie de jouer de la musique, pas de jouer de la perceuse…

Le kit Piezobarrel que j’ai commandé contient un peu de visserie pour pouvoir monter le micro sur un autre instrument si besoin. Le barillet livré avec le Piezo est parfaitement standard et complètement interchangeable avec celui de ma clarinette. Je le monte donc sur l’instrument, visse dessus le Piezobarrel, branche la prise mini-jack du câble de 4m fourni avec le kit, et je me dépêche de brancher l’autre côté du câble – une jack mono standard - sur un ampli.

Si vous en êtes à ce stade, coupez l’alimentation de l’ampli, baissez le volume au max, faites tous vos branchements, et seulement ensuite, vous r’mettez l’son. Vos enceintes vous en seront reconnaissantes, et vos tympans encore plus.

Sur un côté du Piezobarrel, il y a une petite vis qui permet de régler le niveau de sortie du Piezo. Un petit tournevis est fourni avec le kit pour faciliter les réglages. Après avoir joué quelques notes, je décide d’ouvrir la vis à fond, histoire d’envoyer le niveau maxi. Pour la suite de mes expériences, je n’ai pas éprouvé le besoin de baisser le niveau de sortie, mais peut-être changerai-je d’avis plus tard.

Quand on amplifie son instrument pour la première fois comme c’était mon cas, ce qui frappe tout de suite, c’est le fait qu’on est enveloppé par le son. On l’entend comme jamais on ne l’a entendu auparavant, on découvre enfin son propre son, et ce en temps réel.

Car il faut savoir que quand on joue d’un instrument comme la clarinette, le bec est en contact direct avec les incisives de la mâchoire supérieure, et par ce biais une partie des vibrations de l’instrument se répercute dans la boîte crânienne du musicien, qui n’entend pas exactement ce qu’entendent les auditeurs alentour. J’avais découvert le phénomène il y a fort longtemps, en m’enregistrant avec un petit magnéto (l’écoute était alors en différé, c’était très frustrant). Grâce au Piezobarrel et à l’amplification, mon rapport au son produit par la clarinette est totalement bouleversé. Je découvre véritablement un autre son – celui qu’entend mon entourage – et ce en temps réel. C’est une sensation unique… en fait c’est le pied !

À ce stade, plusieurs options sont possibles :

- option 1 : acheter quelques pédales d’effets, un bon ampli, et jouer jusqu’à plus soif : l’idée est séduisante, mais les bonnes pédales coûtent cher. Je casserai peut-être ma tirelire plus tard, mais pour l’heure les options suivantes me tentent plus ;
- option 2 : brancher la clarinette – via le Piezobarrel – sur une carte **Axoloti** (ça, ça me plaît bien) ;
- option 3 : brancher la clarinette – toujours via le Piezobarrel – sur mon PC (qui tourne sous **Ubuntu Studio**), et injecter le son dans quelques logiciels musicaux sympas (bon plan aussi, ça).

Vous l’aurez compris, on va s’intéresser aux options 2 et 3.

2. Branchement du Piezobarrel sur une carte Axoloti

J’avais fait l’acquisition, il y a un peu moins d’un an, d’une carte Axoloti. Cette carte est développée par un ingénieur belge, Johannes Taelman. Elle coûte 65 euros, elle est open source, et c’est une sorte d’hybride entre un synthé et un microcontrôleur. Le site officiel du projet est le suivant : axoloti.com **[7]**.

J’avais fait une présentation de l’Axoloti chez **Deezer**, en juin 2017, dans le cadre du *meetup* **Creative Code Paris [8]**. Pour ceux que ça intéresse, le support de la présentation est disponible sur mon **GitHub [9]**.

Pour faire court : la carte Axoloti se connecte au PC avec un câble USB, et est équipée d’entrées/sorties audio et MIDI. On la programme au moyen d’un *patcher* développé spécifiquement pour elle. Ce *patcher* reprend beaucoup des principes de logiciels - bien connus des musiciens - comme **Pure Data** et **MaxMSP**. Si vous ne voyez pas ce qu’est un *patcher*, ne vous inquiétez pas, on va en voir un exemple dans un instant.

Dotée d’un GPIO comme les cartes **Arduino**, on peut connecter à l’Axoloti à peu près tous les capteurs possibles. La qualité de fabrication est impeccable, et le son est à tomber le... séant par terre.

Le *patcher* qui accompagne la carte est disponible pour **Windows**, **Mac** et **Linux Debian** (paquets .deb). On peut si on le souhaite *uploader* un patch dans la mémoire de la carte (mémoire à ajouter via un connecteur micro-SD), puis la brancher sur une alimentation externe, et transformer ainsi l’Axoloti en un instrument complètement autonome.

J’ai utilisé un temps le *patcher* sur Windows, mais aujourd’hui, je fais mes expérimentations sur Ubuntu Studio (le patcher est identique sur tous les environnements).

Sur la figure 2, vous pouvez voir une photo de la carte Axoloti. On aperçoit les principaux connecteurs : avec à gauche les 2 prises jack In/Out, à droite les 2 prises MIDI In/Out. Vers le milieu de la face avant, de gauche à droite : une sortie stéréo en prise mini-jack, une prise micro-usb, un emplacement pour carte mémoire microSD, et une prise USB. À l’arrière, on aperçoit les connecteurs du GPIO. Sur le côté droit, on aperçoit une prise pour alimentation externe DC 7-15V.



axoloti

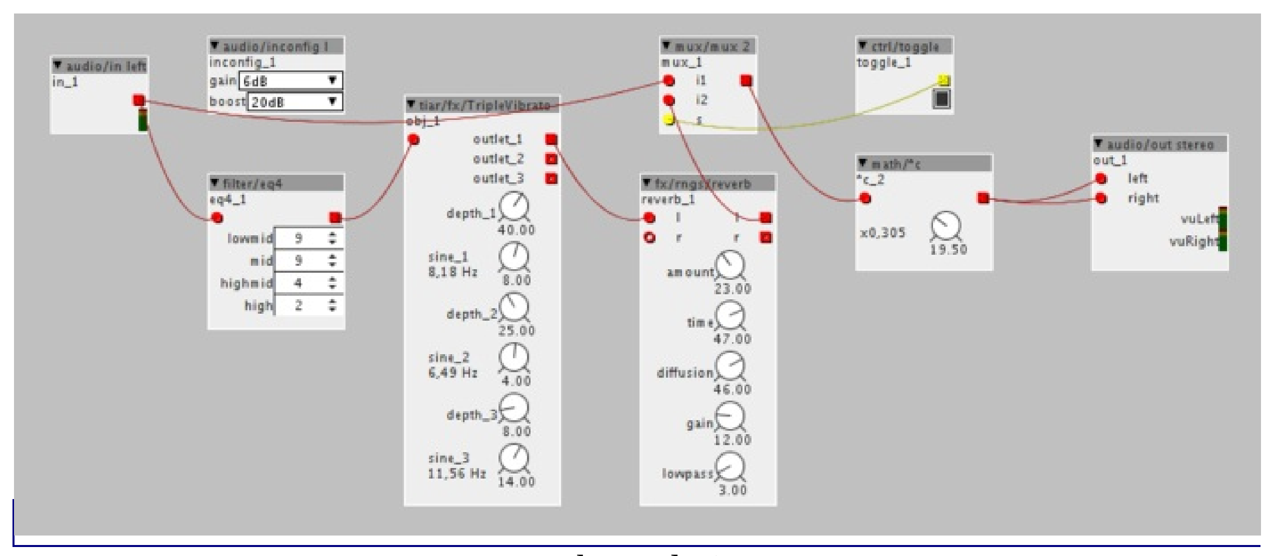
Fig. 2 : Carte Axoloti.

2.1 Quand Piezo parle à Axo

Le raccordement du Piezobarrel avec l’Axoloti se fait via l’entrée jack mono de la carte, et la prise de même format fournie sur le câble audio livré avec le Piezobarrel. Il ne reste plus qu’à créer un patch.

Le *patcher* offre un nombre de fonctionnalités énorme, mais je vais me focaliser ici sur quelques composants permettant de prendre le son de l’entrée ligne et de le renvoyer sur la sortie audio, en le transformant un peu. Un effet de vibrato couplé à un peu de réverbération, ça vous dit ? Et soyons fou, ajoutons aussi un équaliseur 4 bandes, histoire de gonfler un peu le son dans les graves.

Le patch est visible en figure 3.



patch_axoloti_1

Fig. 3 : Patch Axoloti.

Le son transmis par le Piezobarrel est mono, il est réceptionné sur le canal de gauche, d’où l’utilisation du composant audio/in left.

Le composant audio/inconfig l permet de paramétrer le niveau du signal d’entrée (pas obligatoire, mais assez pratique). Avec un gain à 6 db et un *boot* à 20 db, j’obtiens un volume appréciable, parfait pour une écoute au casque notamment, mais je vous recommande de démarrer avec des valeurs moins grandes et de relever les niveaux progressivement pour trouver ce qui vous convient.

Le signal reçu par audio/in left passe par 2 chemins : le premier va vers mux/mux 2 (j’expliquerai pourquoi après). Le second chemin passe par différents effets (équaliseur, vibrato et reverb) avant d’arriver lui aussi sur mux/mux 2.

Vous pouvez définir librement les valeurs du composant filter/eq4, mais une fois le patch lancé, vous ne pouvez pas les modifier en temps réel.

Le composant tiar/fx/TripleVibrato est un des nombreux composants prêts à l’emploi, fournis en standard avec le *patcher*. Les 6 potards (potentiomètres) sont modifiables en temps réel (quand le patch est actif).

Le composant fx/rngs/reverb est un autre composant standard du *patcher*. Ses potards sont aussi modifiables en temps réel.

Le composant mux/mux 2 permet de mixer 2 signaux, le premier étant le signal original reçu par l’Axoloti (le son de la clarinette tout nu), le second signal provient de la palette d’effets (equaliseur, vibrato et reverb). Vous voyez qu’un composant ctrl/toggle est relié au composant mux/mux 2. Il nous permet – en cliquant sur le bouton noir – de désactiver le second signal, si on veut écouter la clarinette sans les effets.

Une fois le patch créé, il faut le sauvegarder, puis cliquer sur le bouton live pour pouvoir le lancer et commencer à jouer.

Dès que vous cliquez sur live, attention à vos enceintes, et surtout attention à vos tympans (et à ceux des copains), car je vous garantis que l’Axoloti peut envoyer du lourd, et que vous pouvez faire du grabuge. Donc, démarrez avec les enceintes réglées au minimum et montez progressivement le son. Et attention au niveau du composant audio/inconfig l que j’ai évoqué plus haut.

L’Axoloti est un formidable synthé. Son *patcher* embarque un grand nombre de

composants. Je ne les ai pas comptés, mais il y en a vraiment beaucoup, avec des oscillateurs, des filtres, des variables, des boutons de toutes sortes, etc. On peut créer des « sous-patchs » que l'on pourra ensuite intégrer dans d'autres patchs, bref, il y a de quoi y passer du temps et développer de beaux projets. Les nombreux exemples fournis avec le *patcher* permettent de se faire une idée assez précise des possibilités, et je vous encourage à tous les tester, en commençant par les exemples de type « séquenceur » qui sont impressionnants.

J'ai présenté ici un exemple assez simple, mais vous pouvez aussi mélanger le son de votre instrument, avec celui émis par différents oscillateurs (le *patcher* en propose plusieurs) pour obtenir des effets encore plus fun

La maîtrise du *patcher* nécessite pas mal de pratique, et donc du temps. Si vous êtes néophyte en matière de *patcher*, vous devriez apprécier le livre de Jan Vantomme, *« Getting started with Axoloti »*, disponible chez leanpub.com **[10]**. Ce livre est une aide précieuse pour s'initier au fonctionnement de l'Axoloti. Le forum du site officiel est aussi une aide précieuse. On y trouve une communauté sympa, très active et pas avare de conseils.

3. Branchement du Piezobarrel sur le PC

Ah là, ça se gâte. Car il ne vous aura sûrement pas échappé que la plupart des PC n'ont plus de prise micro (c'est le cas d'un grand nombre de portables aujourd'hui). Et sur Linux, vous avez intérêt à faire gaffe au type de carte son externe que vous pouvez brancher sur votre PC, car certaines cartes nécessitent l'installation de *drivers* qui n'existent que pour Windows et Mac. Je peux en témoigner, car j'ai une carte son externe USB un peu ancienne qui traîne dans un coin. Je comptais m'en servir pour mes petites expériences, mais j'ai renoncé après avoir tenté en vain de la faire dialoguer avec quelques logiciels tournant sous Ubuntu Studio.

Un ami m'a signalé l'existence d'une page du site linuxmao.org **[11],** et je dois dire que cette page – qui propose des tableaux comparatifs couvrant un grand nombre de cartes sons – m'a aidé à y voir plus clair.

Après avoir lu le commentaire très positif d'un contributeur du site linuxmao, j'ai opté pour une petite interface externe USB à 35 euros, une **Behringer UCG102**, équipée de 2 prises Jack (une entrée guitare, et une sortie casque). Au début de son utilisation (il y a 3 semaines), j'en étais très content. Reconnue tout de suite par Ubuntu Studio, je me suis bien amusé avec, jusqu'à ce qu'elle me lâche il y a 2 jours. En effet, la prise casque est brutalement devenue muette, alors que le son continue à arriver sur l'entrée guitare (je le vois grâce aux potentiomètres de certains logiciels). Le problème est apparu alors que je faisais des tests avec **Chuck** (le sujet de la section suivante). Peut-être ai-je balancé sans le vouloir des fréquences que l'interface externe n'a pas bien digérées. Il est vrai que l'usage premier de l'interface Behringer UCG102, c'est de capter du son, et je l'ai peut-être poussée au-delà de ses limites durant mes petites expériences sonores.

Je vais quand même vous raconter tout ce que j'ai pu faire avec ma petite interface USB, avant qu'elle ne me lâche. Car je l'ai utilisée de 2 manières différentes :

- la première a consisté à injecter le son de la clarinette dans un langage de programmation dédié au son, qui s'appelle Chuck ;
- la seconde a consisté à injecter le son dans quelques softs de musique pour Linux, en particulier dans **Rakarrack**, un simulateur de pédale d'effets, et dans Audacity.

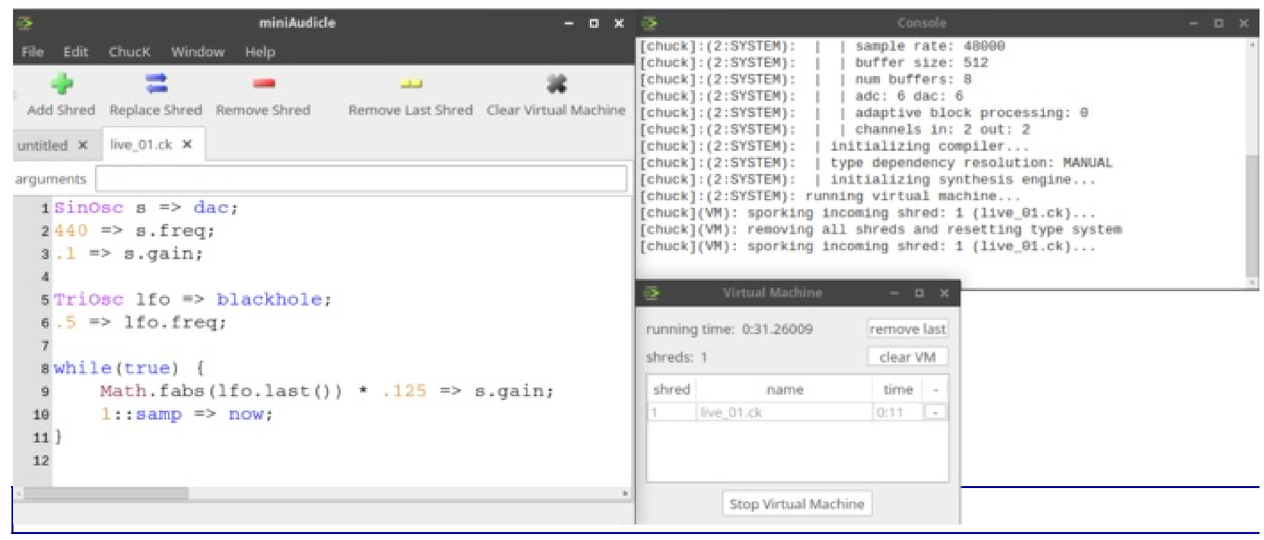
À noter que, chronologiquement, j'avais d'abord testé Rakarrack et **Audacity**, avant de passer à Chuck, mais je trouvais plus sympa de finir cet article avec Rakarrack.

3.1 Une clarinette, ça chuck énormément

Depuis quelques mois, je m'intéresse à un logiciel qui s'appelle Chuck **[12]**. C'est un véritable langage de programmation dédié à la musique électronique. Il est utilisé par certains artistes de la scène électro pour des concerts ébouriffants en mode « live-coding ». Le principal concurrent de Chuck, c'est certainement **Supercollider**, qui est peut-être plus connu en France. J'aime bien Chuck pour sa syntaxe concise, et sa manière unique de gérer certains aspects du son. Le projet a été développé par des chercheurs du CCRMA (*Center for Computer Research in Music and Acoustics)* de l'université Stanford, et il est soutenu par des enseignants de l'université de Princeton. Il semble que le projet ne soit pas très actif depuis 2015, mais il est très complet à ce stade. Il offre une large palette de fonctions, et c'est un super outil pour s'initier à la musique électronique (c'est d'ailleurs pour ça qu'il a été conçu, à la base).

Chuck existe pour Linux, Mac et Windows. On peut utiliser Chuck en ligne de commandes, mais je vous recommande de l'essayer avec **MiniAudicle**, un petit IDE très pratique pour travailler avec Chuck. L'installation de MiniAudicle sous Linux se fait via une procédure d'installation spécifique **[13]**.

Une fois installé, MiniAudicle se présente sous la forme de 3 fenêtres indépendantes : une fenêtre pour la saisie du code, une fenêtre de type console, une fenêtre pour la machine virtuelle qu'il faut démarrer pour pouvoir écouter les scripts écrits en code Chuck (voir figure 4).



chuck_miniaudicle

Fig. 4 : Les trois fenêtres de MiniAudicle.

Dans l’exemple de la figure 4, la machine virtuelle est démarrée et est en train d’exécuter le script de la fenêtre de gauche. Pour lancer un script, il faut cliquer sur le bouton vert en forme de croix qui s’intitule add Shred. Chaque script en cours d’exécution est donc un « shred » pour la machine virtuelle, qui est capable d’en exécuter plusieurs en même temps. On peut ainsi créer des espaces sonores, incluant des nappes, des boucles, et un grand nombre d’effets inclus dans le langage de Chuck (sans compter que l’on peut créer ses propres fonctions, et donc ses propres effets).

Pour vous donner une petite idée de la syntaxe de Chuck, voici un petit script dans lequel on initialise un oscillateur sinusoïdal, qu’on affecte à la sortie audio (le « dac »). On a ensuite une boucle sans fin dans laquelle on initialise 3 sons avec 3 fréquences différentes. Les 2 premiers sons sont joués à tour de rôle, une demi-seconde chacun, et le 3ème son est joué une seconde, avant que la boucle ne redémarre pour un nouveau cycle :

```
SinOsc foo => dac;

while (true) {

   220 => foo.freq;

   .5::second => now;

   440 => foo.freq;

   .5::second => now;
```

```
    880 => foo.freq;
    1::second => now;
}
```

On trouve pas mal de vidéos sur YouTube montrant les possibilités de Chuck pour le *live coding*.

S’il s’agissait seulement de produire du son, l’interface audio externe n’aurait aucun intérêt pour programmer Chuck, mais en ce qui me concerne, je veux injecter du son dans Chuck et voir ce que je peux en tirer.

Si vous constatez que Chuck ne communique pas avec votre interface audio, jetez un coup d’œil dans le menu Edit de MiniAudicle, option Preferences. Vous pourrez réajuster les paramètres Audio Input et Audio Output en fonction de votre configuration.

Pour capter le son envoyé sur l’ordinateur par l’interface audio, c’est très simple. Voici un petit exemple de sketch permettant de capter l’entrée audio (via l’objet adc fourni par Chuck). Une boucle sans fin, permet de jouer instantanément le son reçu, et d’augmenter le gain (volume) au passage.

```
// the patch
adc => Gain x => dac;
2 => x.gain; // on double le volume initial
// infinite time-loop
while( true ) {
    // sur cet exemple très simple, une durée de 10, 100
    //  ou 1000 ms ne fait pas de réelle différence à l’écoute
    100::ms => now;
}
```

On retrouve tous les exemples embarqués avec Chuck sur une page **[14]** administrée par l’Université de Princeton.

Comme Chuck est surtout orienté « live coding », il y a peu d’exemples montrant comment tirer véritablement parti d’une entrée son externe. Par exemple, j’aimerais récupérer la fréquence du son de la clarinette pour lui

appliquer quelques transformations en temps réel... comment faire ça ?

Heureusement, il y a un bon bouquin édité par Manning.com, qui s'intitule « *Programming for Musicians and Digital Artists* » **[15]**. Ce livre est consacré à Chuck, et il donne davantage d'exemples sur le sujet qui m'intéresse. En m'inspirant des exemples du livre, j'ai obtenu quelques effets intéressants que je vous encourage à essayer :

- un effet de type « Octaver » descendant le son en entrée d'une octave :

```
adc => PitShift p => dac;

2 => p.gain;

// forever shifting pitch

while (true) {

   0.5 => p.shift;  // descend le son d'une octave

   1000 :: ms => now;

}
```

Si vous remplacez la valeur 0.5 (de p.shift) par la valeur 2, vous montez le son d'une octave. Vous pouvez tester différentes valeurs intermédiaires entre 0.5 et 2 (et même au-delà de ces limites), les effets obtenus sont très intéressants.

On peut ajouter un peu de réverbération en remplaçant la première ligne par :

```
adc => PitShift p => NRev r => dac;
```

Il y a 3 réverbs différentes que vous pouvez tester : NRev, PCRev, et JCRev.

- un effet de type « Octaver » assez proche du précédent, mais avec une part d'aléatoire :

```
adc => PitShift p => dac;

//1.0 => p.mix;  // paramètre .mix à essayer (on aime ou pas)

2 => p.gain;     // augmentation du volume

while (true) {

   // p.last() renvoie le dernier échantillon sonore sous forme

   //   d'un nombre de type float

   if (p.last() > 0.5){
```

```
        // saut de -1 octave
        0.5 => p.shift;
    } else {
        // saut au hasard entre +/- 1 octave
        Math.random2f(0.5,2.0) => p.shift;
    }

    1000 :: ms => now; // essayer des valeurs comme 1, 10 et 100 ms
}
```

Je voulais faire encore quelques essais en ajoutant un peu de « delay », mais c'est le moment où mon interface audio USB m'a lâché, alors je m'arrête là pour cette partie.

À noter que, de temps en temps, Chuck a tendance à partir en vrille. On s'en rend compte notamment quand le son produit est en retard par rapport au son émis, alors que rien dans le script ne le prévoyait. Peut-être un petit souci du côté de la machine virtuelle... Si ça vous arrive, sauvegardez prestement votre travail, arrêtez la machine virtuelle, fermez Chuck et relancez-le. Cela devrait régler le problème.

Rétrospectivement, je crois bien que c'est à partir de ces derniers tests que l'interface Behringer UCG102 a cessé de fonctionner... ou en tout cas qu'elle a cessé d'émettre du son sur la prise casque. Donc prudence si vous décidez de l'utiliser comme je l'ai fait, en tout cas vous êtes prévenu.

3.2 Une clarinette qui se prend pour une guitare

Ubuntu Studio est fourni avec un nombre conséquent de softs dédiés aux musiciens. Nous allons parler dans cette dernière section de **Jack**, de Rakarrack et de Audacity. Et nous allons découper en tranches le son de la clarinette avec ces softs sympas.

Attention, pour pouvoir utiliser des logiciels comme Rakarrack et Audacity avec une entrée externe, il faut passer par Jack. Je vous laisserai le soin de lire la page de présentation de Jack proposée par Linuxmao **[16]** ; disons, pour faire court, que Jack est le couteau suisse qui va permettre aux différents logiciels musicaux

de se parler, un *patcher* pour logiciels en somme...

Sur Ubuntu Studio, j’utilise Jack au travers de **QjackCtl** (voir figure 5), une petite interface graphique très pratique pour la configuration de Jack. Si vous ne l’avez pas en standard, vous ne devriez pas avoir de mal à l’installer sur votre distribution Linux.

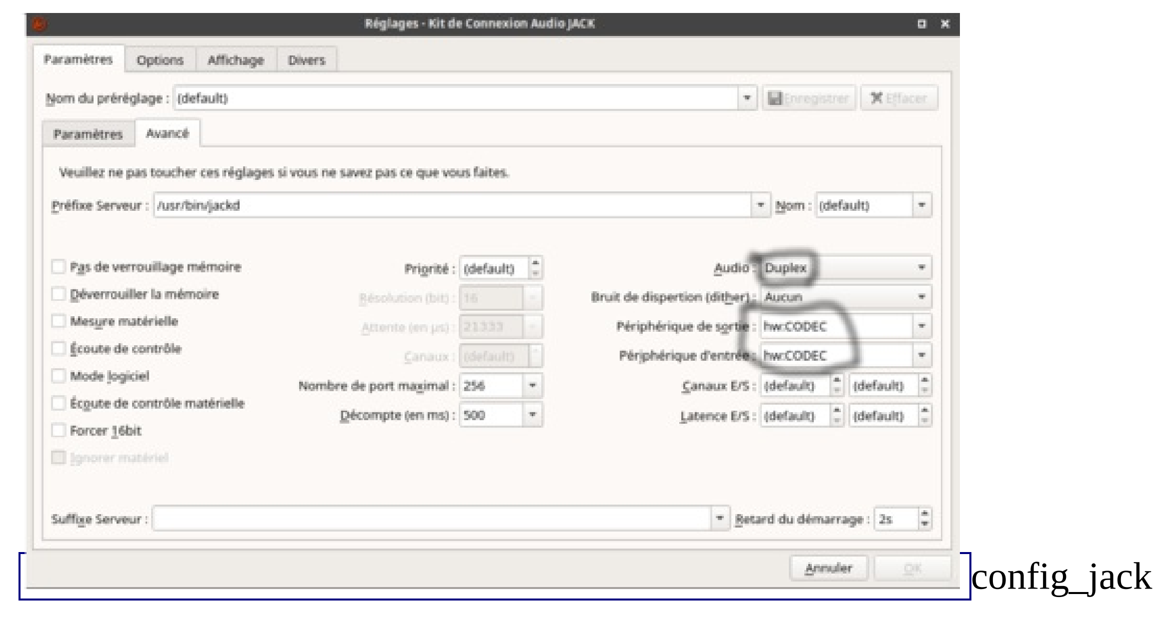


qjackctl

Fig. 5 : Interface de QjackCtl.

La config que je vais expliquer fonctionne avec l’interface audio externe USB – ou plutôt « fonctionnait » avant qu’elle ne me lâche – mais si vous utilisez une autre interface (compatible Linux sans *driver* spécifique), la procédure devrait être identique pour vous.

Branchez l’interface audio USB externe, puis cliquez sur le bouton Réglages de QJackCtl, sélectionnez l’onglet Avancé, et indiquez à droite que vous souhaitez travailler avec une interface USB en entrée comme en sortie (vous verrez la mention hw:CODEC apparaître après cette sélection). Sélectionnez aussi le mode Duplex si pas déjà fait (voir figure 6).

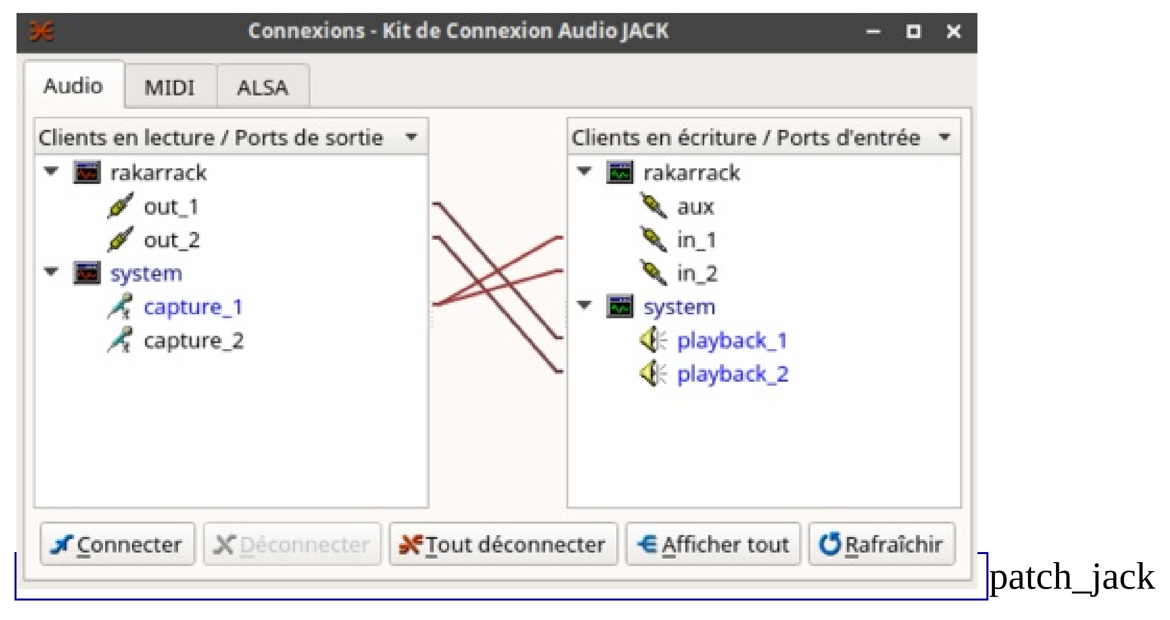


config_jack

Fig. 6 : Réglages de QjackCtl.

N’oubliez pas de cliquer sur OK après avoir modifié les réglages.

Il est temps de démarrer Rakarrack. S’il n’est pas déjà installé, allez fouiller dans les paquets de votre distribution. À partir du moment où Rakarrack est démarré, vous allez pouvoir paramétrer un autre élément de Jack. Pour ce faire, cliquez sur le bouton démarrer de QjackCtl, puis sur le bouton Connecter. Dans la fenêtre qui apparaît, effectuez les branchements comme indiqués en figure 7. On voit le raccordement des entrées et sorties de Rakarrack et du système. Le système, c’est en définitive votre interface audio externe USB, si elle est branchée (sinon c’est le *hardware* de votre PC).

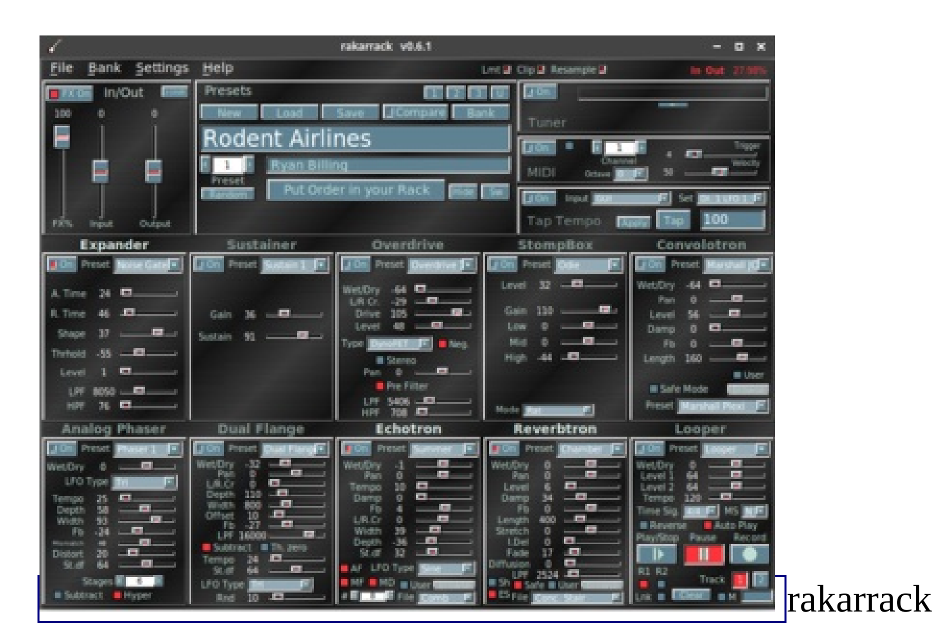


patch_jack

Fig. 7 : Connexions Jack.

Je rappelle que le son du Piezobarrel est mono, et qu’il arrive sur le canal de gauche. Le canal de gauche correspond à capture_1, j’envoie donc capture_1 sur les 2 entrées (gauche et droite) de Rakarrack.

Les branchements virtuels sont terminés. Revenez sur l’interface de Rakarrack, cliquez sur le bouton Fx On en haut à gauche (voir figure 8).



rakarrack

Fig. 8 : Interface de Rakarrack.

Sélectionnez quelques effets en cliquant sur leurs boutons On respectifs, par exemple un coup d'Expander, un peu d'Echotron, un peu de Reverbtron, soufflez dans la clarinette, et amusez-vous !!

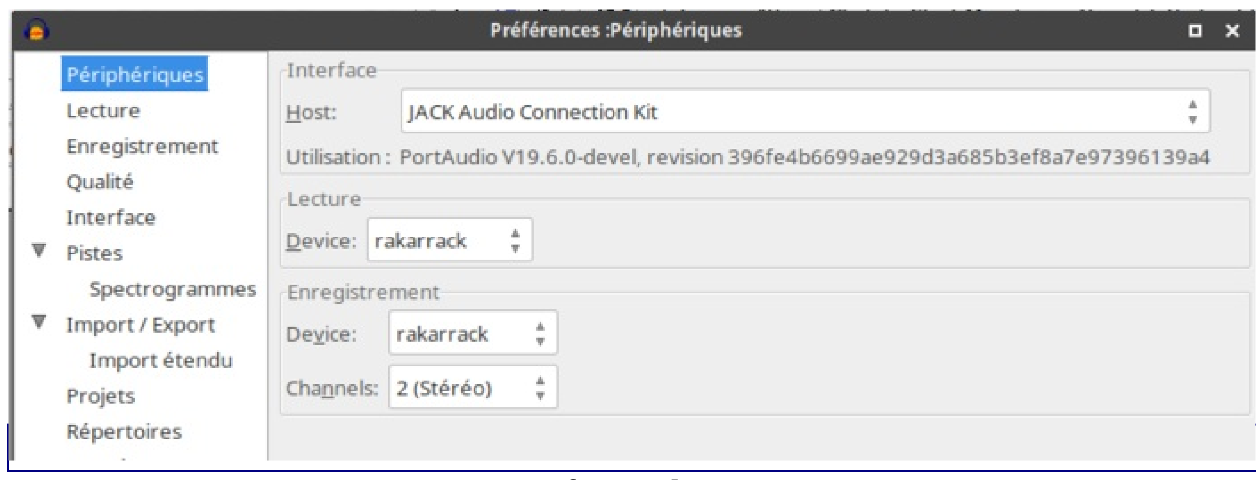
On le voit sur l'interface de la figure 8, Rakarrack propose un grand de nombre de paramètres pour chaque effet, alors dites-vous que vous êtes parti pour quelques heures d'expérimentation, peut-être même quelques dizaines d'heures. Pensez à faire une pause de temps en temps.

Un ami musicien m'a dit récemment que la MAO était hautement addictive. Je crois bien qu'il a raison.

Au fait si vous avez du mal à vous arrêter, faites comme moi, grillez votre interface audio (ne me demandez pas le mode d'emploi, je n'ai rien compris).

Je vous propose pour finir d'enregistrer le son, transformé par Rakarrack, sur le logiciel Audacity.

Pour que cela fonctionne, on a un petit paramétrage à faire dans Audacity. Allez dans le menu Édition, option Préférences. Sélectionnez ensuite l'option Périphériques, et paramétrez l'interface comme indiqué en figure 9.



config_audacity

Fig. 9 : Paramétrage d'Audacity.

Attention, vous allez très certainement obtenir un son sursaturé au départ, vous devrez ajuster le volume dans Rakarrack, mais ça, vous le verrez à l’usage.

En parlant de saturation, il m’est arrivé à plusieurs reprises de voir le son saturer brutalement, quelquefois au bout de 15 minutes de jeu. Après quelques recherches, je suis tombé sur le conseil suivant **[17]**:

« Sur les distributions qui ne sont pas tournées vers la MAO il est fort à parier que le noyau Lowlatency ne soit pas installé ou bien que lors d'une mise à jour de votre distribution le noyau soit remplacé par un noyau "Généric" , dans ce cas il faut changer le noyau pour un lowlatency et peut-être un : **sudo dpkg-reconfigure -p high jackd** *»*

J’ai appliqué ce conseil, mais je vous avoue que je manque de recul pour vous dire si cela a été efficace. En tout cas j’ai appliqué cette technique au moins une semaine avant que l’interface audio externe ne me lâche, donc je ne pense pas qu’il y ait un lien de cause à effet (… encore que…).

Conclusion

Avec l’arrivée d’un micro comme le Piezobarrel – d’excellente qualité pour un prix très raisonnable – il me semble que les instruments à vent traditionnels (bois et cuivres) peuvent connaître un véritable renouveau. Ce petit micro permet aux instrumentistes à vent qui le souhaitent, de s’embarquer dans le grand chaudron de la musique électro, et d’expérimenter librement en interfaçant leur instrument préféré avec des outils qui étaient jusque-là plutôt réservés aux claviéristes et aux guitaristes.

Pour les musiciens qui auraient du mal à se projeter dans ces nouveaux moyens d’expression, c’est peut-être le moment de découvrir – ou redécouvrir – les travaux de John Surman **[18]**.Ce saxophoniste et clarinettiste est un précurseur dans l’utilisation de *loops* et de pédales d’effets avec des instruments à vent. Ses albums *« Road to Saint Ives »* et *« Private City »* sont des références en la matière.

Le Piezobarrel est encore peu connu en France, de même que l’Axoloti. J’espère que ce modeste article contribuera à les faire connaître un peu plus chez nous. Je souhaite aux créateurs de ces deux projets beaucoup de succès, et je ne les remercierai jamais assez de m’avoir aidé – sans le savoir – à renouer avec ma bonne vieille clarinette.

Références

[1] Wikipédia, « *Clarinette* » : https://fr.wikipedia.org/wiki/Clarinette

[2] Wikipédia, « *MIDI* » : https://fr.wikipedia.org/wiki/Musical_Instrument_Digital_Interface

[3] HAYES B., vidéo de présentation du Piezobarrel : https://www.youtube.com/watch?v=scEvGbbFO2g

[4] HAYES B., vidéo du Piezobarrel transformé par une pédale d'effets pour guitare électrique : https://www.youtube.com/watch?v=CsvQp4UQGjQ

[5] HAYES B., vidéo du Piezobarrel transformé différentes pédale d'effets : https://www.youtube.com/watch?v=GOJnEuWJc_E

[6] Piezobarrel : http://piezobarrel.com

[7] Axoloti : http://www.axoloti.com/

[8] Creative Code Paris : https://www.meetup.com/fr-FR/CreativeCodeParis/

[9] Présentation de l'Axoloti : https://github.com/gregja/meetup_ccp_axoloti

[10] VANTOMME J., « *Getting started with Axoloti* » : https://leanpub.com/getting-started-with-axoloti

[11] LinuxMAO : www.linuxmao.org/tiki-index.php?page=Cartes son

[12] Chuck : http://chuck.cs.princeton.edu

[13] Installation de MiniAudicle : https://raw.githubusercontent.com/ccrma/miniAudicle/miniAudicle-1.3.3/notes/README.linux

[14] Exemples de Chuck : http://chuck.cs.princeton.edu/doc/examples/

[15] KAPUR A. et al., « *Programming for Musicians and Digital Artists* » : https://www.manning.com/books/programming-for-musicians-and-digital-artists

[16] Présentation de Jack : http://linuxmao.org/Jack

[17] Résoudre les problèmes de saturation : http://linuxmao.org/Jack#Craquements_et_autres_bruits_parasites_avec_Jack

[18] Wikipédia, « John Surman » : https://fr.wikipedia.org/wiki/John_Surman